AMENDMENTS TO THE CLAIMS

1. (currently amended) In a computer system, a method for implementing and using a filter object, the method comprising:

providing the filter object, the filter object including a state, and the filter object being represented by:

an output equation for generating an output of the filter object, and

a state equation for updating the state of the filter object;

implementing the filter object; and

determining an output of the filter object based on an input to the filter object~~using the filter object~~ in a first dynamically typed text-based programming environment,~~the using the filter object~~ determining the output of the filter object including:

receiving [[an]]the input at the filter object,

identifying a first state of the filter object,

splitting up the input into a first input and a second input,

performing a first operation by processing the output equation to determine [[an]]a first output of the filter object based on the first input of the filter object and the first state of the filter object,

performing a second operation by processing the state equation to determine a second state of the filter object based on the first input of the filter object and the first state of the filter object,

retaining the second state of the filter object in the first dynamically typed text-based programming environment, [[and]]

making the second state available after the output equation of the filter object is processed, and

performing a third operation by processing the output equation to determine a second output of the filter object based on the second input of the filter object and the second state of the filter object.

2. (previously presented) The method of claim 1 wherein the filter object retains a final state obtained after processing the input of the filter object.

3. (previously presented) The method of claim 2 wherein the final state retained in the filter object is used as an initial state for processing a second input of the filter object.

4. (previously presented) The method of claim 1 further comprising the step of resetting a state of the filter object retained in the filter object.

5. (previously presented) The method of claim 1 further comprising the step of presetting a state of the filter object retained in the filter object.

6. (canceled)

7. (canceled)

8. (previously presented) The method of claim 1 wherein the filter object generates code for implementing a corresponding filter algorithm.

9. (previously presented) The method of claim 1 wherein the filter object generates code to implement a test bench or a filter analysis.

10. (previously presented) The method of claim 8 wherein the filter object executes in a simulation environment, and the generated code is executed outside a context of the simulation environment in which the filter object executes.

11. (previously presented) The method of claim 10 wherein the generated code is in a textual language.

12. (previously presented) The method of claim 10 wherein the generated code is in a graphical description language.

13. (previously presented) The method of claim 8 wherein the filter object executes in a simulation environment, and the generated code is executed within a context of the simulation environment in which the filter object executes.

14. (previously presented) The method of claim 8 wherein the generated code is executed in an environment separate from a computer system used for a simulation of the filter object, including an embedded system implementation.

15. (previously presented) The method of claim 14 wherein the generated code is in a textual language.

16. (previously presented) The method of claim 14 wherein the generated code is in a graphical description language.

17. (previously presented) The method of claim 14 wherein the generated code is suitable for use with a software implementation, the software implementation being adapted for use of the generated code on at least one of a general purpose processor, a digital signal processor, and a programmable computer architecture.

18. (previously presented) The method of claim 14 where the generated code is suitable for use with a hardware implementation, the hardware implementation including use with at least one of a Field Programmable Gate Array (FPGA), Complex Programmable Logic Device (CPLD), and Application Specific Integrated Circuit (ASIC) device, the generated code being written in hardware description language.

19. (previously presented) The method of claim 8 wherein the code is written in a high-level programming language.

20. (previously presented) The method of claim 8 wherein the code is written in a low-level machine or assembly language.

21. (currently amended) A computer-implemented method for generating an output of a system in response to an input to the system, the method comprising:

implementing the system using a dynamically typed text-based programming environment; and

using the system in the dynamically typed text-based programming environment, the system:

performing a first operation to determine ~~determining~~ [[an]]a first output of the system based on [[an]]a first input to the system and a first state of the system;

performing a second operation to determine ~~determining~~ a second state of the system based on the first input to the system and the first state of the system; [[and]]

retaining the second state of the system in the dynamically typed text-based programming environment so that the second state is available after the output of the system is determined; and

performing a third operation to determine a second output of the system based on the second input to the system and the second state of the system.

22. (previously presented) The method of claim 21 further comprising specifying equations that the system processes to generate the output of the system using the input to the system and a state of the system.

23. (previously presented) The method of claim 21 further comprising controlling a state of the system retained in the memory.

24. (previously presented) The method of claim 23 wherein a state of the system retained in the memory is reset to provide a zero initial state to the system.

25. (previously presented) The method of claim 23 wherein a state of the system retained in the memory is set to a particular value entered by a user.

26. (previously presented) The method of claim 21 wherein a state of the system retained in the memory includes a final state obtained by processing the input to the system.

27. (previously presented) The method of claim 21 wherein a state of the system is an initial state of the system for processing the input of the system.

28. (currently amended) A computer readable medium holding instructions executable in a computer that provides a dynamically typed text-based programming environment, the instructions comprising:

providing an object, the object being an instance of a class;

performing a first operation to determine ~~determining~~ [[an]]a first output of the object based on [[an]]a first input to the object and a first state of the object;

performing a second operation to determine ~~determining~~ a second state of the object based on the first input to the object and the first state of the object;

retaining the second state of the object in the dynamically typed text-based programming environment; [[and]]

making the second state available after determining the output of the object; and

performing a third operation to determine a second output of the object based on a second input to the object and the second state of the object.

29. (original) The medium of claim 28 further comprising the step of instantiating the object from the class.

30. (original) The medium of claim 28 wherein the object includes an adaptive filter object.

31. (previously presented) The medium of claim 30 wherein the adaptive filter object includes an adapting algorithm that the adaptive filter implements.

32. (original) The medium of claim 28 wherein the object includes a discrete time filter object.

33. (previously presented) The medium of claim 28 further comprising controlling properties of the object including a state of the object.

34. (original) The medium of claim 33 wherein the state of the object is reset to zero.

35. (previously presented) The medium of claim 28 further comprising inheriting a state property corresponding to a state of the object from an abstract class.

36. (previously presented) The medium of claim 28 further comprising providing the class with methods which operate on the object of the class.

37. (currently amended) A system for implementing a filter object, the system comprising:

a processor configured to process:

an output equation of the filter object processed in a first operation to determine [[an]]a first output of the filter object based on [[an]]a first input to the filter object and a first state of the filter object, and processed in a second operation to determine a second output of the filter object based on a second input to the filter object and a second state of the filter object;

a state equation of the filter object processed in a third operation to determine [[a]]the second state of the filter object based on the first input to the filter object and the first state of the filter object; and

a memory for retaining the second state of the filter object in a dynamically typed text-based programming environment so that the second state is available after the output equation is processed.

38. (canceled)

39. (previously presented) The system of claim 37, wherein the second state retained in the memory is used as a state of the filter object in processing a next input of the filter object.

40. (canceled)

41. (canceled)

42. (previously presented) The system of claim 37, wherein a state of the filter object retained in the memory is reset to provide a zero initial state.

43. (previously presented) The system of claim 37, wherein a state of the filter object retained in the memory is set to a particular value entered by a user.

44. (previously presented) The method of claim 1, wherein the filter object operates on a sample-by-sample, block-by-block or frame-by-frame basis.

45. (previously presented) The method of claim 21, wherein the system operates on a sample-by-sample, block-by-block or frame-by-frame basis.

46. (previously presented) The medium of claim 28, wherein the object operates on a sample-by-sample, block-by-block or frame-by-frame basis.

47. (previously presented) The system of claim 37, wherein the input of the filter object comprises a single piece of sample data, a sequence of sample data or multiple sequences of sample data.